Simplifying Entity Resolution on Web Data with Schema-agnostic, Non-iterative Matching

Vasilis Efthymiou, George Papadakis, Kostas Stefanidis, Vassilis Christophides

Simplifying Entity Resolution on Web Data with Schema-agnostic, Non-iterative Matching

Vasilis Efthymiou
ICS-FORTH
Greece
vefthym@ics.forth.gr

George Papadakis
Univ. of Athens
Greece
gpapadis@di.uoa.gr

Kostas Stefanidis
Univ. of Tampere
Finland
kostas.stefanidis@uta.fi

Vassilis Christophides
INRIA-Paris & Univ. of Crete
France & Greece
vassilis.christophides@inria.fr

***Abstract*—Entity Resolution (ER) aims to identify different descriptions in various Knowledge Bases (KBs) that refer to the same entity. ER is challenged by the *Variety*, *Volume* and *Veracity* of descriptions published in the Web of Data. To address them, we propose the MinoanER framework that fulfills *full automation* and support of *highly heterogeneous* entities. MinoanER leverages a token-based similarity of entities to define a new metric that derives the similarity of neighboring entities from the most important relations, indicated only by statistics. For high efficiency, similarities are computed from a set of *schema-agnostic blocks* and processed in a non-iterative way that involves four *threshold-free heuristics*. We demonstrate that the effectiveness of MinoanER is comparable to existing ER tools over real KBs exhibiting *low heterogeneity* in terms of entity types and content. Yet, MinoanER outperforms state-of-the-art ER tools when matching *highly heterogeneous* KBs.**

I. Introduction

Entity Resolution (ER) is a core task for applications integrating data that pertain to entities (e.g., persons, places). In the Web of Data, ER allows for interlinking data that describe the same real-world entity, but are located in different Knowledge Bases (KBs) [1], [2]. Two are the core ER problems: (a) *how can we effectively compute the similarity of Web entities*, and (b) *how can we efficiently resolve descriptions of entities published by different KBs*. Both problems are challenged by the *Variety*, *Volume* and *Veracity* of the Web of Data. Variety is mainly due to the high diversity of entity types described by numerous vocabularies in different domains covered by KBs. Volume concerns both the number of KBs and the number of published entity descriptions. Veracity stems from various forms of inconsistencies, noise or errors in entity descriptions, due to the limitations of the automatic extraction techniques or of the crowd-sourced contributions.

The above Big Data properties call for novel ER frameworks that relax a number of assumptions underlying the state-of-the-art methods. The most important one is related to the notion of similarity that better characterizes *entity descriptions* in the Web of Data - we define an entity description to be a URI-identifiable set of attribute-value pairs, where values can be literals, or the URIs of other descriptions, this way forming an *entity graph*. Clearly, Variety renders inapplicable all schema-based similarity measures, which compare specific attribute values. We thus argue that similarity evidence of entities can be obtained by looking at the bag of strings contained in descriptions, regardless of the corresponding attributes.

As this **value-based similarity** of entity pairs may still be weak, due to high heterogeneity, we need to consider additional sources of matching evidence; for instance, the **similarity of neighboring entities**, which are interlinked via various semantic relations. In state-of-the-art systems, like [3], [4] and [5], this is done through an *iterative process* that relies on *domain knowledge* for the equivalence of relations between neighboring entities. In contrast, we argue that no iterative process is needed to assess the impact of neighbor similarity in a candidate pair, while an estimation of which entity relations in this neighborhood are important to consider can be guided by simple data statistics. Another assumption that needs relaxation is the use of schema-based blocking for addressing Volume by reducing the candidate pairs to similar descriptions. Most existing works, like [3] and [5], rely on blocking keys known in advance, a requirement that is unrealistic for loosely structured and highly heterogeneous entities published on the Web. We argue that *schema-agnostic* blocking methods (e.g., Token Blocking [6]) should be preferred, as they achieve high recall without considering the attribute names.

Overall, the main requirements for Web-scale ER are: *(i)* do not rely on a given schema, *(ii)* do not rely on domain experts for aligning relations and matching rules, *(iii)* avoid late convergence through a non-iterative process. Currently, no existing framework simultaneously accomplishes all these requirements. To cover these requirements, we present the MinoanER framework, which leverages a schema-agnostic set of blocks to define a new metric assessing the similarity of a set of neighboring entity pairs linked via important relations to the entities of a candidate pair. Rather than requiring an *a priori* knowledge of the entity types or of their correspondences, we rely on *simple statistics over two KBs to recognize the most important entity relations* involved in their neighborhood, as well as, *the most distinctive attributes that could serve as names of entities* beyond the `rdfs:labels`, which are not always available in descriptions. Both similarity metrics can be computed using exclusively block statistics (e.g., block size).

Additionally, MinoanER involves a specific number of predefined heuristic steps (H1-H4), instead of the data-driven convergenece of existing systems ([3], [4], [5]). First, it identifies matches based on their name (H1). This is a very effective method that can be applied to all descriptions, regardless of their values or neighbor similarity, by *automatically specifying*

distinctive names of entities from data statistics. Then, the value similarity is exploited to find matches with many common and infrequent tokens, i.e., strongly similar matches (H2). When value similarity is not high, nearly similar matches are identified based on both value and neighbors' similarity using a *threshold-free rank aggregation function* (H3) as opposed to existing works that combine different matching evidence into an aggregated score. Finally, *reciprocal evidence* of matching is exploited as a verification of the returned results: only entities mutually ranked in the top positions of their unified ranking lists are considered matches (H4).

We demonstrate the benefits of MinoanER through an experimental comparison against the state-of-the-art methods over 4 established benchmark datasets that involve real KBs.

II. Related Work

Value-based similarities, like Jaccard, usually assess the descriptions similarity based on their attribute values. Our value similarity is a variation of ARCS [6], [7] that drops any schema information and considers descriptions as a bag of words. Compared to ARCS, we focus more on the *number than the frequency of common tokens between two descriptions*.

Relational similarities additionally consider neighbor similarity. For example, [3] and [5] consider the similarity of "compatible" neighbors, linked with pre-aligned relations, while [4] considers only neighbors linked via relations with similar labels. *Our approach does not aggregate different similarities in one score; instead, it uses a disjunction of the different evidence coming from the values, neighbors and names of the descriptions. The most important neighbors are detected automatically from dataset statistics.*

Based on the nature of the matching decision, ER can be characterized as *pairwise* or *collective*. In pairwise ER (e.g., [8]), we only need to know the value similarity of descriptions to decide if they match. Collective ER (e.g., [9]) iteratively updates the matching decision for entities by dynamically assessing the similarity of their neighbors. *We propose a static collective approach*, in which all sources of similarity are assessed only once per candidate pair through a specific number of steps.

In more detail, [3] starts with seed matches having identical entity names. Then, it propagates the matching decisions on the compatible neighbors of existing matches. Unique Mapping Clustering is applied for detecting matches. First, it places all pairs into a priority queue, in decreasing (relational) similarity. At each iteration, the top pair is considered a match, if none of its entities has been already matched and their similarity exceeds a threshold t. For every new matched pair, the similarities of the neighbors are recomputed and their position in the priority queue is updated. The process ends when the top pair has a lower similarity than t. [4] differs by considering as compatible neighbors those connected with relations having similar names, which rarely holds in the Web of Data. [5] is a similar approach, introducing the following heuristic: if two matched descriptions e_1, e_1' are connected via aligned relations r, r' and all their entity neighbors via r, r', except e_2, e_2', have been matched, then e_2, e_2' are also considered matches. Three are the main differences of our work to [3], [4] and [5]. First, our matching process iterates over a set of blocks, instead of the initial KBs. Second, we employ statistics to automatically discover distinctive entity names and important relations. Third, we exploit different sources of matching evidence (values, names and neighbors) to statically identify candidate matches already from blocking.

Finally, [10] uses a probabilistic model to identify matches, based on previous matches and the functional nature of entity relations. A relation is considered functional if, for a source entity, there is only one destination entity. If $r(x, y)$ is a function in a KB and $r(x, y')$ a function in another KB, then y, y' are considered matches. *Unlike our approach, PARIS cannot deal with structural heterogeneity.*

III. MinoanER Matching Process

We now describe the heuristics comprising the non-iterative matching process of MinoanER in the order they are applied.

Name Heuristic (H1). The matching evidence of H1 relies on *entity names*. As such, we consider the literal values of the k attributes in every description with the highest importance. We define the *importance* of a predicate p in a KB $\mathcal{E}$ as the harmonic mean of its *support*, i.e., the portion of entities in $\mathcal{E}$ that contain p, and *discriminability*, i.e., the ratio between the distinct objects associated with p and the entities that contain p in their description. H1 treats the entire entity names as blocking keys to create a set of blocks, B_N. Every block with one entity from each input KB indicates a pair of matching entities. This way, H1 assumes that *two entities match, if they, and only they, have the same name*. All candidates matched by H1 are not examined by the remaining heuristics.

Value Heuristic (H2). The rationale in H2 is that *two entities match, if they, and only they, share a common token, or if they share many infrequent tokens.* Basically, H2 identifies pairs of descriptions with high *value similarity*. For two entities $e_i \in \mathcal{E}_1, e_j \in \mathcal{E}_2$, this similarity is defined as: $valueSim(e_i, e_j) = \sum_{t \in tokens(e_i) \cap tokens(e_j)} \frac{1}{log_2(EF_{\mathcal{E}_1}(t) \cdot EF_{\mathcal{E}_2}(t) + 1)}$, where $EF_{\mathcal{E}}(t) = |\{e_l | e_l \in \mathcal{E} \wedge t \in tokens(e_l)\}|$ stands for "Entity Frequency", i.e., the number of entities in $\mathcal{E}$ having token t in their values. For high efficiency, H2 applies Token Blocking to the input KBs, yielding a set of blocks B_T. Then, it goes through the blocks of every entity e_i of the smaller in size KB that hasn't been matched by H1, to derive its value similarity with all co-occurring entities of the other KB. From all co-occurring entities, it keeps e_j, which corresponds to the highest value similarity, v_{max}. If $v_{max} \geq 1$, H2 considers the pair (e_i, e_j) to be a match. Matches identified by H2 will not be considered in the sequel.

Rank Aggregation Heuristic (H3). This heuristic identifies further matches for candidates whose value similarity is low (v_{max}<1), yet their neighbor similarity could be relatively high. In this respect, the order of candidates rather than their absolute similarity values are used. In essence, H3 defines the *neighbor similarity* of two entity descriptions $e_i \in \mathcal{E}_1, e_j \in \mathcal{E}_2$ as: $neighborNSim(e_i, e_j) = \sum_{ne_i \in topNneighbors(e_i), ne_j \in topNneighbors(e_j)} valueSim(ne_i, ne_j)$,

where $topNneighbors(e_i)$ stands for the best neighbors of e_i, i.e., those associated with it through one of the N relations with the maximum importance score. On this basis, H3 goes through every description that has not been matched yet and sorts the entities co-occurring with it in the blocks of B_T in two lists: one in decreasing order of value similarity and one in decreasing non-zero neighbor similarity. Then, it aggregates the two lists by considering the normalized ranks of their elements: assuming the size of a list is K, the first candidate gets the score K/K, the second one $(K-1)/K$, while the last one $1/K$. Overall, each co-occurring entity of e_i takes a score equal to the weighted summation of its normalized ranks in the two lists, as determined through the trade-off parameter $\theta \in (0, 1)$: the value similarities are weighted with θ and the neighbor ones with with 1-θ. At the end, we keep for e_i, its top-1 candidate match e_j, i.e., the one with the highest aggregate score. Intuitively, H3 *matches e_i with e_j, when there is no better candidate for e_i than e_j.*

Reciprocity Heuristic (H4). It aims to clean the matches identified by H1, H2 and H3 by exploiting *reciprocity*, i.e., based on the rationale that two entities are unlikely to match, when one of them does not even consider the other to be a candidate for matching. Intuitively, H4 aims to improve the precision of our algorithm by enforcing the requirement that *two entity descriptions match, only if both of them "agree" that they are likely to match.* H4 essentially iterates over every matched pair $< e_i, e_j >$ that has been detected by the above heuristics and discards it if e_i does not include e_j in its top-K value or neighbor similarity candidates, or vice versa.

In a nutshell, every heuristic can be formalized as a function that receives a pair of entities and returns true (T) if the entities match according to the heuristic's rationale, or false (F) otherwise, i.e.,: $Hn : \mathcal{E}_1 \times \mathcal{E}_2 \rightarrow \{T, F\}$. In this context, we formally define the MinoanER matching process as follows:

Definition 1. The **non-iterative matching** of two KBs $\mathcal{E}_1$, $\mathcal{E}_2$, denoted by the Boolean matrix $M(\mathcal{E}_1, \mathcal{E}_2)$, is defined as a filtering problem of the pruned disjunctive blocking graph G: $M(e_i, e_j) = (\mathsf{H1}(e_i, e_j) \vee \mathsf{H2}(e_i, e_j) \vee \mathsf{H3}(e_i, e_j)) \wedge \mathsf{H4}(e_i, e_j)$.

The time complexity of MinoanER is dominated by the comparisons in the input blocks $B_N \cup B_T$. In the worst-case, this results in one computation for every pair of entities, i.e., $O(|\mathcal{E}_1| \cdot |\mathcal{E}_2|)$. In practice, though, we bound the number of computations by removing excessively large blocks that correspond to highly frequent tokens (e.g., stop-words). Following [6], this is carried out by Block Purging, which ensures that the resulting blocks involve two orders of magnitude fewer comparisons than the brute-force approach, without any significant impact on recall.

IV. Experimental Evaluation

Experimental Setup. All experiments were performed using Java 8 on a server with Intel(R) Xeon(R) E5-2630 v4 @ 2.20GHz and 64 GB RAM, runnng Ubuntu 16.04.2 LTS. Preliminary experiments have indicated that the following parameter configuration yields robust performance for MinoanER across all datasets: K=15 (candidate matches per entity from values and neighbors), N=3 (most important relations per entity), k=2 (most distinct attributes per KB whose values serve as names), and θ=0.6 (trade-off between value- vs neighbor-based candidates).

TABLE I
Dataset statistics.

	Restaurant	Rexa-DBLP	BBCmusic-DBpedia	YAGO-IMDb
$\mathcal{E}_1$ entities	339	18,492	58,793	5,208,100
$\mathcal{E}_2$ entities	2,256	2,650,832	256,602	5,328,774
$\mathcal{E}_1$ triples	1,130	87,519	456,304	27,547,595
$\mathcal{E}_2$ triples	7,519	14,936,373	8,044,247	47,843,680
$\mathcal{E}_1$ av. tokens	20.44	40.71	81.19	15.56
$\mathcal{E}_2$ av. tokens	20.61	59.24	324.75	12.49
$\mathcal{E}_1/\mathcal{E}_2$ attributes	7 / 7	114 / 145	27 / 10,953	65 / 29
$\mathcal{E}_1/\mathcal{E}_2$ relations	2 / 2	103 / 123	9 / 953	4 / 13
$\mathcal{E}_1/\mathcal{E}_2$ types	3 / 3	4 / 11	4 / 59,801	11,767 / 15
$\mathcal{E}_1/\mathcal{E}_2$ vocab.	2 / 2	4 / 4	4 / 6	3 / 1
Matches	89	1,309	22,770	56,683

Datasets. We use 4 benchmark datasets (Table I) with entities from real KBs commonly used in the literature. All KBs contain relations between the described entities. *Restaurant*[1], a popular dataset published by OAEI[2], contains restaurants descriptions and their addresses from two different KBs. *Rexa-DBLP*[3] contains publications descriptions and their authors. The ground truth contains matches between both publications and authors. *BBCmusic-DBpedia* [11] contains descriptions of musicians, bands and their birthplaces, from BBCmusic and the BTC2012 version of DBpedia[4]. In our experiments, we consider only entities appearing in the ground truth, as well as their immediate in- and out-neighbors. *YAGO-IMDb* [10] contains descriptions of movie-related entities (e.g., actors, directors, movies) from YAGO and IMDb[5].

Baselines. We compare MinoanER against four state-of-the-art methods, [3], [10], [4], [5], and a custom baseline method, BSL. BSL receives the same input as MinoanER, i.e., the sets of blocks B_N and B_T, and compares every pair of co-occurring descriptions. The resulting similarities are processed by Unique Mapping Clustering. Unlike MinoanER, BSL disregards all evidence from neighbors, relying exclusively on value similarity. Yet, it optimizes its performance with respect to F1 through: *(i)* The schema-agnostic representation of the values in every entity. BSL uses token n-grams for this purpose, $n \in \{1, 2, 3\}$, representing every resource by the token uni-/bi-/tri-grams in its values. *(ii)* The weighting scheme that assesses the importance of every token. We consider TF and TF-IDF weights. *(iii)* The similarity measure: Cosine, Jaccard, Generalized Jaccard and SiGMa [3]. *(iv)* The similarity threshold that prunes the entity pairs processed by Unique Mapping Clustering. We use all thresholds in $[0, 1)$ with a step of 0.05. We consider 420 different configurations for BSL per dataset, reporting the one with the highest F1.

[1] http://oaei.ontologymatching.org/2010/im
[2] http://oaei.ontologymatching.org
[3] http://oaei.ontologymatching.org/2009/instances
[4] datahub.io/dataset/bbc-music, km.aifb.kit.edu/projects/btc-2012/
[5] www.yago-knowledge.org/, www.imdb.com/

TABLE II
BLOCK STATISTICS.

	Restaurant	Rexa-DBLP	BBCmusic-DBpedia	YAGO-IMDb
$\|B_N\|$	83	15,912	28,844	580,518
$\|B_T\|$	625	22,297	54,380	495,973
$\|\|B_N\|\|$	83	$6.71{\cdot}10^7$	$1.25{\cdot}10^7$	$6.59{\cdot}10^6$
$\|\|B_T\|\|$	$1.80{\cdot}10^3$	$6.54{\cdot}10^8$	$1.73{\cdot}10^8$	$2.28{\cdot}10^{10}$
$\|\mathcal{E}_1\| \cdot \|\mathcal{E}_2\|$	$7.65{\cdot}10^5$	$4.90{\cdot}10^{10}$	$1.51{\cdot}10^{10}$	$2.78{\cdot}10^{13}$
Precision	4.95	$1.81{\cdot}10^{-4}$	0.01	$2.46{\cdot}10^{-4}$
Recall	100.00	99.77	99.83	99.35
F1	9.43	$3.62{\cdot}10^{-4}$	0.02	$4.92{\cdot}10^{-4}$

Results. Table II reports the performance of the blocks used by BSL and MinoanER. The number of comparisons in token blocks ($||B_T||$) is at least 1 order of magnitude larger than those of name blocks ($||B_N||$), even if the latter may involve more blocks ($|B_N|>|B_T|$ over YAGO-IMDb). In fact, the comparisons suggested by names seem to depend linearly on the number of input descriptions, whereas the comparisons suggested by tokens seem to depend quadratically on that number. Nevertheless, the overall comparisons in $B_T \cup B_N$ are at least 2 orders of magnitude lower than the Cartesian product $|E_1| \cdot |E_2|$, even though recall is consistently higher than 99%. Yet, both precision and F1 remain rather low.

Table III reports the performance of MinoanER and the baselines. For every method, we report precision, recall and F1 with respect to the descriptions in the first KB appearing in the ground truth. [10] is openly available, so we ran it on Rexa-DBLP and BBCmusic-DBpedia. For the remaining tools, we report their performance from the original publications - [5] is openly available, but without instructions.

Table III shows that MinoanER offers competitive performance when matching KBs with few attributes and entity types, even if it requires no domain-specific input, while achieving the best performance over highly heterogeneous KBs. It achieves 100% F1 in Restaurant, which is 3% higher than SiGMa, 9% higher than PARIS, and ∼20% higher than LINDA and RiMOM. BSL also achieves perfect F1, due to the strongly similar matches in this dataset. In Rexa-DBLP, MinoanER also outperforms all existing ER methods. It is 2% better than SiGMa in F1, 4.6% better than PARIS, 20% better than RiMOM, and 6% better than BSL. Given that BBCmusic-DBpedia is the most heterogeneous dataset with respect to schema and values, PARIS struggles to identify the matches, with BSL performing significantly better, but still poorly in absolute numbers. In contrast, MinoanER succeeds in identifying 89% of matches with 91% precision, achieving a 90% F1. In YAGO-IMDb, MinoanER achieves similar performance to SiGMa (91% F1), with more identified matches (91% vs 85%), but lower precision (91% vs 98%). Compared to PARIS, its F1 is 1% lower, due to 3% lower precision, despite the 1% better recall. Finally, BSL exhibits the worst performance, due to the very low value similarity of matches in this KB.

Comparing the performance of MinoanER (Table III) to that of its input blocks (Table II), precision raises by several orders of magnitude at the cost of slightly lower recall. The lower recall is caused by missed matches with very low value and neighbor similarities, whose portion is larger for BBCmusic-DBpedia and YAGO-IMDb.

TABLE III
EVALUATION OF MINOANER COMPARED TO EXISTING METHODS.

		Restaurant	Rexa-DBLP	BBCmusic-DBpedia	YAGO-IMDb
SiGMa [3]	Prec.	99	97	-	98
	Recall	94	90	-	85
	F1	97	94	-	91
LINDA [4]	Prec.	100	-	-	-
	Recall	63	-	-	-
	F1	77	-	-	-
RiMOM [5]	Prec.	86	80	-	-
	Recall	77	72	-	-
	F1	81	76	-	-
PARIS [10]	Prec.	95	93.95	19.40	94
	Recall	88	89	0.29	90
	F1	91	91.41	0.51	**92**
BSL	Prec.	100	96.57	85.20	11.68
	Recall	100	83.96	36.09	4.87
	F1	**100**	89.82	50.70	6.88
MinoanER	Prec.	100	96.74	91.44	91.02
	Recall	100	95.34	88.55	90.57
	F1	**100**	**96.04**	**89.97**	90.79

V. CONCLUSION

To resolve highly heterogeneous Web entities, MinoanER[6] relies on schema-agnostic similarity metrics that consider the content and neighbors of the entities. For high efficiency, these similarities are extracted from a set of blocks and processed by a non-iterative process that involves four threshold-free heuristics. The experimental results show that our approach achieves at least equivalent performance with state-of-the-art ER tools over KBs exhibiting low heterogeneity, but outperforms them to a significant when matching highly heterogeneous.

REFERENCES

[1] K. Stefanidis, V. Christophides, and V. Efthymiou, "Web-scale blocking, iterative and progressive entity resolution," in *ICDE*, 2017.

[2] V. Christophides, V. Efthymiou, and K. Stefanidis, *Entity Resolution in the Web of Data*. Morgan & Claypool Publishers, 2015.

[3] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani, "Sigma: simple greedy matching for aligning large knowledge bases," in *KDD*, 2013.

[4] C. Böhm, G. de Melo, F. Naumann, and G. Weikum, "LINDA: distributed web-of-data-scale entity matching," in *CIKM*, 2012.

[5] C. Shao, L. Hu, J. Li, Z. Wang, T. L. Chung, and J. Xia, "Rimom-im: A novel iterative framework for instance matching," *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 185–197, 2016.

[6] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl, "Meta-blocking: Taking entity resolutionto the next level," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1946–1960, 2014.

[7] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, and T. Palpanas, "Parallel meta-blocking for scaling entity resolution over big heterogeneous data," *Inf. Syst.*, vol. 65, pp. 137–157, 2017.

[8] L. Kolb, A. Thor, and E. Rahm, "Dedoop: Efficient deduplication with hadoop," *PVLDB*, vol. 5, no. 12, pp. 1878–1881, 2012.

[9] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *TKDD*, vol. 1, no. 1, 2007.

[10] F. M. Suchanek, S. Abiteboul, and P. Senellart, "PARIS: probabilistic alignment of relations, instances, and schema," *PVLDB*, vol. 5, no. 3, pp. 157–168, 2011.

[11] V. Efthymiou, K. Stefanidis, and V. Christophides, "Big data entity resolution: From highly to somehow similar entity descriptions in the web," in *IEEE Big Data*, 2015.

[6]Source code is available at: http://csd.uoc.gr/~vefthym/minoanER.